VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

IS administrace účtů na katedrálních databázových systémech

IS for Administration of Database Management Systems

2013 **Martin Zwierzyna**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Martin Zwierzyna**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: IS administrace účtů na katedrálních databázových systémech
IS for Administration of Database Management Systems

Zásady pro vypracování:

Katedra informatiky poskytuje výuku pěti předmětů v oblasti databázových systémů. Kromě toho je každoročně několik desítek bakalářů, diplomantů a studentů doktorského studia pracujících v této oblasti. Pro tyto účely má katedra k dispozici několik serverů, na kterých jsou nainstalované databázové systémy. Cílem této práce je vytvořit IS, který bude spravovat užívatelské účty na těchto databázových systémech.

1. Nastudovat způsoby administrace jednotlivých databázových systémů.
2. Nastudovat přihlašení užívatele do systému pomocí LDAP a definování práv jednotlivých uživatelů.
3. Navrhnout, naimplementovat a otestovat IS pro administraci dostupných databázových systémů.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Peter Chovanec**

Datum zadání: 16.11.2012
Datum odevzdání: 07.05.2013

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Dne: *1.5.2013*

……..……
podpis studenta

Poděkování

Rád bych poděkoval *Ing. Peteru Chovancovi* za odbornou pomoc a konzultaci při vytváření této bakalářské práce.

Abstrakt

Motivem pro vytvoření této práce byla neefektivita dosavadního řešení administrace účtů na katedrových databázových systémech. Pro řešení tohoto problému je potřeba vytvořit informační systém, který by umožňoval jednoduchou a hlavně efektivní správu účtů na libovolných katedrových databázových serverech. Tím je myšlena skutečnost, že systém bude podporovat více databázových serverů najednou díky dynamickému vytváření instancí. Pro přidání nového databázového systému bude potřeba pouze doimplementovat předem vytvořené metody a funkce, které si vývojář vygeneruje z předem připraveného rozhraní. Vytváření účtu se bude provádět na základě tří události a těmi jsou – žádost, jednotlivé vytvoření uživatele, import uživatelů ze systému Edison. Přihlašování bude řešeno pomocí kontroly přihlašovacích údajů na školním adresářovém serveru LDAP. Součástí systému bude také správa platností jednotlivých účtů, kdy systém bude upozorňovat na propadlé účty. Dále bude také systém rozesílat e-maily o aktivitách vyvolaných na tomto informačním systému – vytvoření uživatelského konta, zamítnutí žádosti.

Klíčová slova

Informační systém, databázový systém, databázové uživatelské systém LDAP, šifrování, technologie ASP.Net, e-mailová notifikace.

Abstract

The aim of the thesis was replacing of present non-effective administration of database accounts on the Department of Computer Sciences with information system, which would allow simple and fast accounts administration on selected servers of the department. The system will allow dynamic creation of DBMS instances and so support administration of several database servers. In case of adding new type of database server, there will be a need to add previously created functions and methods, which will be generated by the developer using the prearranged interface. A new account will be created based on three events which are: based on the request, individual user creation and multiple creation for users exported from Edison. Login will be solved using the login information check on school adressing server LDAP. The System will also provide the administration of validities of individual accounts and it will notify about the expired accounts. In case of operations like creation of an user, denial of a request, the system will send e-mails owners concerned.

Key words

Information system, database system, database accounts, LDAP system, Encryption, ASP.Net, e-mail notification.

Seznam použitých zkratek

Zkratka	Anglický význam	Český význam
ASLR	Address Space Layout Randomization	zabezpečení
ASP	Active server pages	skriptovací platforma
CLR	Common Language Runtime	běhové prostředí, spouští kód
COM	Component Object Model	objektové orientovaný systém pro vytváření binárních softwarových komponent
DES	Data Encryption Standard	bloková šífra založena na DES, ale aplikována třikrát
DN	Distinguished Name	unikátní identifikace záznamu
JIT	Just In Time	překlad programu, který je spuštěn
LDAP	Lightweight Directory Access Protocol	protokol pro ukládání a přístup k datům na adresářovém serveru
RODC	Read Only Domain Controller	součást Active Directory, sloužící k zabezpečení
Server IIS	Server Internet Information Services	webový server s kolekcí rozšiřujících modulů
XML	Extensible Markup Language	značkovací jazyk

Obsah

1 Úvod

Databáze je určitá uspořádaná množina informaci (neboli dat) uložená na paměťovém médiu. V širším smyslu jsou součástí databáze i softwarové prostředky, které umožňují manipulaci s uloženými daty a přístup k nim. Tento software se v české odborné literatuře nazývá systém řízení báze dat (SŘBD). Běžně se označením databáze (v závislosti na kontextu) myslí jak uložená data, tak i software (SŘBD).

V dnešní době se lidé setkávají s informačními systémy na mnoha místech, mnohdy ani o tom nemusí vědět. Příkladem informačního systému může být kartotéka, telefonní seznam, kniha došlé pošty nebo účetnictví. Systém nemusí být nutně automatizovaný pomocí počítačů, ale může být i v papírové podobě. Informacemi jsou myšlena sdělení, která odstraňují nejistotou nebo nevědomost, daty míníme jakékoli zaznamenané poznatky či fakta. Jako zvláštní pojem zde vystupuje také znalost představující zobecnění určité části reality. Informaci je možno také chápat jako data s nějakým přidaným významem (data + význam). Informace je údaj, ke kterému si člověk přiřadí význam. Již dlouho je jasné, že hospodářství vyspělých zemí netáhnou jen hmotné výrobky, ale také informace, znalosti a nové technologie. To si uvědomují i podniky a instituce, což napomáhá k rozvoji IS.

Důležité je tedy ujasnit si základní rozdíl mezi databází a informačním systémem. Databáze představuje uložení dat na určitém místě, kdežto informační systém je už konkrétní řešení daného problému.

Proto jsou součástí výuky oboru Informační a komunikační technologie také databázové předměty, některé jsou zaměřené na práci s databázemi samotnými a v jiných se učí studenti navrhovat vlastní informační systém na základě znalostí získaných z předmětů zaměřených na databázové systémy. V databázových předmětech se studenti učí základním znalostem pro úspěšné zavedení a správu databáze, kdežto v předmětech zaměřených na vývoj informačních systému se student učí navrhovat systém, vytvářet funkční analýzu, vytvářet datovou analýzu, používat návrhové vzory až po samotnou implementaci informačního systému, ať už jako desktopovou nebo internetovou aplikaci.

Tyto předměty jsem rovněž absolvoval, a proto jsem se také rozhodl vybrat pro tuto bakalářskou práci právě téma se zaměřením na vývoj informačního systému. Volba byla motivována nejen zájmem o databázovou problematiku, ale hlavně snahou o vytvoření systému, který bude lidem nápomocný a může jim případně ulehčit spoustu práce.

2 Dosavadní řešení administrace účtů

V současné době se na Vysoké škole báňské vyučuje několik předmětů zaměřených na problematiku databázové logiky (Úvod do databázových systémů, Databázové a Informační systémy, apod.). Obecně se na těchto hodinách ukazují a řeší různé úlohy a problémy z širokého okruhu databázové logiky. Aby bylo možné tyto úlohy a problémy společně se studenty řešit, je potřeba, aby měl každý student možnost připojit se na některý ze školních databázových serverů. Jiným případem může byt žádost studenta o vytvoření účtu na určitém databázovém serveru např. z důvodu projektu v předmětu, který nespadá pod databázové předměty. V tomto případě nemá student jinou možnost, než e-mailem poslat takovou žádost některému pedagogovi, který má přístup ke studentem žádané databázi. Tento způsob komunikace a žádosti je ale zbytečně zdlouhavý proces, který jsem sám musel absolvovat (v mém případě se ještě navíc zkomplikoval spatně přidělenými právy mého účtu). Avšak v této době neexistuje žádný nástroj, který by umožňoval vytvářet účty přímo ze seznamů studentů v jednotlivých skupinách, uvedených v systému Edison. Proto je potřeba tyto účty vytvářet ručně a to pomocí standartních příkazů, ať už v Microsoft SQL Server Management studio, nebo Oracle SQL Developer. Vytvoření uživatele však neobnáší pouze jeden příkaz, ale množinu příkazů kdy je uživateli vytvářena databáze, login a především práva. Dalším problémem může být zjišťování příčin nefunkčností systému/účtů. Jak jsem již výše zmiňoval, u mé žádosti nastal problém s připojením ke školnímu serveru Oracle. Důvodem byly špatně nastavena práva pro připojení k databázi, proto je potřeba, aby systém také umožňoval sledovat nebo nejlépe nastavovat práva pro připojení. Dále se také účty musí na konci semestru hromadně mazat, což je v případě ručního mazání zdlouhavá operace. Proto je potřeba vytvořit informační systém, který by všechny tyto problémy určitým způsobem ulehčil a automatizoval.

2.1 Databázové systémy

V mé práci jsem se rozhodl řešit problematiku dvou serverů, které jsem za dobu svého studia využíval k absolvování databázových předmětů. Jedná se o servery Microsoft SQL Server a Oracle Database 11g. Informační systém bude ale podporovat daleko více databázových serverů (MySQL, DB2, Postgres apod), avšak v současné době se vyučuje pouze na Microsoft SQL Server a Oracle Database 11g a proto bylo cílem práce se na tyto dva systémy při vývoji zaměřit.

2.1.1 Microsoft SQL Server

Microsoft SQL Server [8] představuje ucelenou sadu technologií a nástrojů připravených pro podnikové řešení, které uživatelům pomáhají vytěžit z informací maximální hodnotu a to při nejnižších celkových nákladech na vlastnictví. Nabízí vysokou úroveň výkonnosti, dostupnosti i zabezpečení, produktivnější nástroje pro správu, vývoj a získání lepšího přehledu o podniku díky samoobslužným funkcím pro bussiness intelligence.

Obrázek 2.1: Logo Microsoft SQL Server

Shrnutí

- Ucelená informační platforma pro aplikace všech velikostí
- Známé spravované samoobslužné nástroje pro BI
- Podpora rozsáhlých datacenter a datových skladů
- Příležitosti k budování a rozšiřování aplikací v cloudu
- Integrace s aplikační platformou Microsoft

Hlavní důvody, proč právě zvolit Microsoft SQL Server

1. Správci databází na platformě Microsoft dokáží provozovat více důležitých databází než v případě databázi Oracle
2. Microsoft SQL Server nabízí 99,99995% dostupnost
3. Microsoft SQL Server se vyznačuje nejvyšším zabezpečením ze všech důležitých databázových platforem
4. Microsoft SQL Server dosahuje nejlepších výsledků v benchmarkových testech TCP-E
5. Microsoft SQL Server přináší úsporu ročních nákladů na správu ve výší 460 % na databázi v porovnání s řešením Oracle

2.1.2 Oracle Database 11g

Servery společnosti Oracle jsou navrženy tak, aby poskytovaly rekordní výkon, zjednodušenou správu, vysokou dostupnost a efektivitu šetřící náklady. Tyto špičkové systémy zahrnují integrovanou virtualizaci, správu cloudu a systému a jsou optimalizovány pro provoz systémů Oracle Solaris, Oracle Linux, Oracle VM a Oracle Enterprise Manager Ops Center. Kromě toho tyto špičkové systémy podporují software pro aplikace a řešení společností Oracle a dalších výrobců.

Databáze Oracle Database 11g [7] pomáhá zákazníkům snížit náklady na IT a poskytovat kvalitnější služby umožněním konsolidace do databázových cloudů a propracovaných systémů jako Oracle Exadata a Oracle Database Appliance. Jejích rychlost, spolehlivost, bezpečnost a snadná

spravovatelnost je prověřená pro všechny typy databázového pracovního vytížení, včetně podnikových aplikací, datových skladů a rozsáhlých datových analýz.

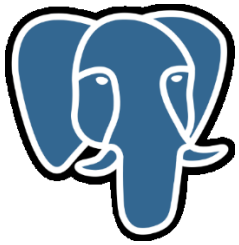
Obrázek 2.2: Logo Oracle Database 11g

Hlavní důvody, proč právě zvolit Oracle Database 11g

1. **Vysoká dostupnost** – s databází Oracle Database 11g a architekturou s maximální dostupností společnosti Oracle lze omezit náklady spojené s výpadky ochranou podniku před všemi běžnými příčinami plánovaných i neplánovaných prostojů, včetně lidských chyb
2. **Správa uložiště** – databáze Oracle Database 11g umožňuje nenákladnou správu úložišť díky automatizaci procesů, minimalizace nákladných operaci I/O, komprimaci dat a maximální využití vrstvených uložišť pro všechny podnikové databáze
3. **Database Security** – společnost Oracle nabízí širokou kolekci řešení zabezpečení, které zajišťuje utajení dat, ochranu před ohrožením ze strany interních pracovníku a splnění zákonných požadavků
4. **Databázové cloudy** – společnost Oracle nabízí širokou kolekci softwarových a hardwarových produktů a služeb pro veřejné, soukromé a hybridní cloudy

2.1.3 PostgreSQL

Mnohdy nazýváno také nazýván Postgres. Je to objektově-relační databázový systém. Je vydáván pod licencí typu MIT, tudíž se jedná o free a open source software. Jako u mnoha jiných open source programů, PostgreSQL [11] není vlastněn jedinou firmou, ale na vývoji se podílí globální komunita vývojářů a firem. PostgreSQL je primárně určen pro Linux, neboli obecně unixové systémy, ale existují i balíčky určené pro platformu win32. Zabudovaný jazyk je PL/pgSQL a připomíná procedurální jazyk PL/SQL známý z Oracle. PostgreSQL podporuje C, C++, Java a Common Lisp. Tento systém podporuje funkce, indexy i triggery. Využívá jej například Yahoo!, MySpace, či Skype.



Obrázek 2.3: Logo ProgreSQL

2.1.4 MySQL

MySQL [12] je databázový systém, vytvořený švédskou firmou MySQL AB, v současné době jej vlastní firma Sun Microsystes, která je dceřinou společností Oracle Corporation. MySQL je multiplatformní databáze. Pro svou implementovatelnost, výkon a především díky tomu, že se jedná o volně šiřitelný software, v současné době má vysoký podíl na používaných databázích. MySQL bylo od

počátku optimalizováno na rychlost, a to i za cenu některých zjednodušení. Má jen jednoduché způsoby zálohování, a až donedávna nepodporovalo pohledy, triggery a uložené procedury. Tyto vlastnosti jsou doplňovány v posledních letech. Velmi oblíbená je kombinace Linux, MySQL, PHP a Apache jako základní software webového serveru.

Obrázek 2.4: Logo MySQL

2.1.5 DB2

IBM DB2 [5] je moderní relační databázový systém vyvíjený společností IBM. Jedná se o multiplatformní databázi s pokročilými možnosti kontroly dat, jejich využívání a ochrany. V rodině DB2 existují 3 produkty, které jsou si velmi podobné, nicméně rozdílné především hardwarovým řešením, na kterém operují. Konkrétně jsou to: DB2 for LUW (Linux, Unix a Windows), DB2 for z / OS (mainframe), a DB2 for iSeries (dříve OS/400). DB2 LUW produkt běží na mnoha distribucích systému Linux a UNIX distribucí, jako je Red Hat Linux, SUSE Linux, AIX, HP / UX a Solaris, a většina systému Windows. DB2 také pohání IBM InfoSphere Warehouse, což je v podstatě DB2 LUW s DPF (Database Partitioning Feature), masivní paralelní share-nothing architektura datového skladu.

Obrázek 2.5: Logo DB2

3 Použité technologie

3.1 Microsoft Windows Server 2008

Jelikož se chystám řešit vývoj informačního systému pomoci technologie ASP.Net, systém Windows Server bude z dále uvedených důvodů jasnou volbou.

Windows Server je název operačního systému řady Windows NT vyvíjený společností Microsoft. Byl vydán v roce 2008 a je určen pro použití jako server v počítačové síti. Systém sdílí stejný kód důležitých částí se systémem Windows Vista (od SP1).

Windows Server 2008 vychází ze stejného kódu jako Windows Vista (jádro Windows NT 6.0 kernel), se kterou sdílí mnoho ze své funkcionality a architektury. Automaticky tak těží z výhod spojených s vývojem Windows Vista, jako je přebudovaný síťový modul (IPv6, nativní podpora bezdrátových sítí nebo zvýšení rychlosti a bezpečnosti), lepší podpora instalačních obrazů, spouštění a zálohování, širší možnosti diagnostiky, monitoringu a záznamu událostí serveru, lepší bezpečnostní prvky (Bitlocker, ASLR, RODC, vylepšený Windows Firewall), .NET Framework 3.0, vylepšení jádra a správy pamětí a procesů.

Windows Server 2008 je první operační systém vydaný s Windows PowerShellem, nové technologie Microsoftu v oblasti příkazové řádky a skriptovacích technologií. PowerShell je založen na objektově orientovaném programování a verzi .NET Framework 2.0. Obsahuje více než 120 utilit pro systémovou administraci, jednotnou syntaxi a pojmenovávání a vestavěnou schopnost pracovat se správou dat typu registru Windows, úložiště certifikátů nebo např. WMI (Windows Management Instrumentation. Skriptovací jazyk PowerShellu byl navržen speciálně pro administraci IT a může nahradit jak cmd.exe, tak Windows Scripting Host.

3.2 ASP.Net

Pro vývoj tohoto systému jsem si vybral technologii ASP.Net [9] a to z důvodu možnosti využití webové služby. Díky této technologie lze řešit funkčnost systém pomocí webových stránek. Proto bude tento systém dostupný z prostředí internetu a nebude jej potřeba instalovat.

ASP.Net je technologie umožňující použití služeb nezbytných pro vytváření a případné zavádění podnikových webových aplikací. Tato technologie přináší vývojáři nový model programování a infrastruktury zajišťující bezpečnější a stabilnější aplikace, které mohou komunikovat s jakýmkoliv prohlížečem nebo zařízením.

Technologie ASP.Net je součástí Microsoft .NET Framework, které usnadňuje vývoj aplikací ve vysoce distribuovaném prostředí sítě Internet. Rozhraní .NET Framewrok zahrnuje společný běhový jazykový modul (CLR – Common Language Runtime), který poskytuje služby jako jsou správa podprocesů, správa paměti či zabezpečení kódu. Rozhraní také zahrnuje knihovnu tříd .NET Framework, což je objektově orientovaná kolekce typů, které může vývojář využít k vytváření vlastních aplikací.

3.2.1 Výhody

Zabezpečení – ASP.Net poskytuje výchozí schémata autorizace a ověřování pro webové aplikace. Tato schémata může vývojář jednoduše doplňovat, odebírat či nahrazovat podle svých potřeb.

Výkon – ASP.Net je ve skutečnosti zkompilovaný kód běžící na serveru. Na rozdíl od prostředí ASP (Active Server Pages) může tato technologie využívat výhody služeb časné vazby, kompilace JIT (Just-in-time), nativní optimalizace a služeb mezipaměti pro zvýšení výkonu.

Snadné zavedení – Vývojářem vytvořený projekt lze na server instalovat pouhým zkopírováním publikovaných souborů bez nutnosti restartovaní serveru a to i v případě pouhého nahrazování kompilovaného kódu.

Flexibilní ukládání výstupu do mezipaměti – ASP.Net umožňuje ukládat obsah stránek, části stránek případně celé stránky do mezipaměti v závislosti na potřebě aplikace. Položky, které jsou uloženy v mezipaměti mohou být závislé na souborech či jiných položkách v mezipaměti, nebo mohou být aktualizovány na základě jejich platnosti.

Mobilní zařízení – ASP.Net přímo podporuje jakýkoliv prohlížeč v libovolném zařízení. Vývojáři používají ke komunikaci s novými mobilními zařízeními stejné techniky k programování, jako u klasických internetových prohlížečů dnešních stolní počítačů.

Sledování a ladění – ASP.Net nabízí služby pro sledování, které slouží k ladění na úrovni aplikace i na úrovni stránky. Vývojář si může vybrat, zdali chce tyto informace zobrazit na stránce, nebo na úrovni aplikace promocí nástroje pro zobrazení sledování. Tato technologie podporuje jak místní, tak i vzdálené ladění pomocí ladících nástrojů rozhraní .Net Framework a to během vývoje i při provozu aplikace. Pokud je aplikace uvedena do provozu, mohou být trasovací příkazy ponechány v provozním kódu, aniž by byl ovlivněn výkon.

Kompatibilita s existujícími ASP aplikacemi – Technologie ASP a ASP.Net lze spustit souběžně na webovém serveru IIS, aniž by docházelo k vzájemnému rušení mezi těmito aplikacemi. V případě instalace technologie ASP.Net není možno poškodit již běžící aplikaci ASP. Tato technologie zpracovává pouze soubory s příponou ASPX. Soubory s příponou ASP budou nadále zpracovány modulem ASP.

3.2.2 Důvody pro použití technologie ASP.Net

Zvýšení výkonu a škálovatelnosti

- **Kompilované provádění** – Technologie ASP.Net je značně rychlejší, než běžné prostředí ASP a zároveň zachovává model okamžité aktualizace prostředí ASP. Není vyžadován žádný explicitní kompilační krok, ASP.Net automaticky rozpozná kteroukoliv změnu, v případě potřeby poté dynamicky zkompiluje soubory a uloží kompilované výsledky k opakovanému použití pro další požadavky.
- **Stav relace webové serverové farmy** – stav relace ASP.Net nabízí možnost sdílení dat v počítačích webové serverové farmy. Tímto získává uživatel možnost pro více požadavků využít přístup k více serverům webové serverové farmy, a přesto mu nadále zůstane plný přístup k datům dané relace.

- **Bohaté možnosti ukládání výstupu do mezipaměti –** možnost ukládání výstupu do mezipaměti technologie ASP.Net značně navyšuje výkon a rozšiřitelnost aplikace. Na stránce se povolí ukládání výstupu do mezipaměti a díky tomu prostředí ASP.Net spustí stránku jednou a před odesláním výsledku uživateli ji uloží do paměti. Pokud bude stejnou stránku požadovat jiný uživatel, odešle prostředí ASP.Net již výsledek uložený v paměti, aniž by musel stránku znovu vytvářet.

Zvýšená spolehlivost

- **Ochrana proti nevracení pamětí, zablokování a selhání** – ASP.Net automaticky rozpoznává chyby a to jako zablokování nebo nevrácenou paměť a následně provádí zotavení. Tím je zajištěna trvalá dostupnost aplikace. Pokud je například zjištěna nevrácená paměť, technologie ASP.Net automaticky spustí novou kopii pracovního procesu ASP.Net a nasměruje nové požadavky na nový proces. Když starý proces dokončí zpracování požadavků, které čekají na vyřízení, bude následně odstraněn a nevrácená paměť bude uvolněna.

Snadné zavedení

- **Bezobslužné zavádění aplikací –** pomocí technologie ASP.Net lze aplikaci nainstalovat pouhým zkopírováním na daný server, bez nutnosti restartu. Nastavení konfigurace je poté uloženo v souboru XML aplikace.
- **Dynamická aktualizace spuštěné aplikace –** technologie ASP.Net umožnuje aktualizaci zkompilovaných částí aplikace, bez nutnosti vypínání či restartování celého serveru, na rozdíl od komponent běžného modelu COM, kde bylo vyžadováno restartování serveru bezprostředně po ukončení aktualizace.

Nové modely aplikací

- **Webové služby XML –** webové služby XML jsou primárně určeny pro komunikaci aplikací a ke sdílení dat v síti Internet bez ohledu na operační systém nebo programovací jazyk. Technologie ASP.Net zjednodušuje vystavení a volání webových služeb XML.

Tabulka 3.1: Webové služby XML

Standard	Použití ve službách XML vytvářených pomocí technologie ASP.Net
XML	Textový formát, který se využívá pro komunikaci s webovými službami XML používajícími protokol SOAP. Pro komunikaci s webovými službami XML pomocí protokolů HTTP-GET a http-POST se jazyk XML používá právě pro kódování odpovědi.
SOAP	Jde o protokol pro výměnu zpráv založený na jazyce XML. Používá se pro komunikaci mezi webovými službami XML a klienty.
WSDL	Popis výměny zpráv, které může webová služba XML interpretovat pro komunikaci s klientem.
XSD	Univerzální systém typů umožňující definování typů dat a jejich předávání mezi platformami. Jazyk XSD definuje typy dat pro soubor XML zapouzdřený v rámci zprávy SOAP odeslané z webové služby XML.
Application/x-www-form-urlencoded	Typ MIME používaný ke kódování parametrů pro adresu URL. Používá se pro kódování parametrů požadavků pro webové služby XML používající protokoly HTTP-GET a HTTP-POST.

3.3 Databázový systém

Jelikož potřebná databáze má obsahovat pouze 4 tabulky, nebude potřeba nějakých složitých a časově náročných operací. Proto jsem jako primární databázový systém vybral SQLite. Tento systém je na rozdíl od databází založených na principu klient-server, které jsou spuštěný jako samostatný proces, realizován pouze malou knihovnou, kterou stačí připojit k informačnímu systému. Každou databázi představuje právě jeden soubor s koncovkou *.s3db. Toto dělá ze SQLite lehce přenositelný a také velice rychlý systém. Pro možnost implementace bylo potřeba importovat SQLite.dll knihovnu se správnou verzí .NET jakou má vyvíjený projekt. Tyto aspekty vedly právě ke zvolení SQLite, oproti jiným systémům má výhodu v jednoduchosti, v možnosti aplikace prakticky bez instalace či v rychlosti.

3.3.1 Příprava databázového systému

Pro správnou funkčnost celého IS bylo potřeba připravit tento databázový systém a vytvořit tabulky podle předem vytvořené datové analýzy. Pro případ mazání jsem při nastavování vazeb mezi tabulkami přidal ještě příkaz ON DELETE CASCADE, který mi zajistí smazání všech záznamů spojených s primárním klíčem odkazované tabulky v případě, kdy dojde ke smazání záznamu s tímto primárním klíčem.

4 Analýza systému

Cílem této práce je tedy vytvořit systém, který by umožňoval správu účtů na školních databázových systémech.

4.1 Funkční analýza

4.1.1 Role systému

Jako první je třeba si uvědomit, které osoby budou tento systém obsluhovat.

Role systému:

- Student
- DB Admin
- SYS Aadmin

Student bude mít možnost posílat žádosti na vytvoření účtu. U této žádosti bude muset uvést důvod žádosti a také datum konce platnosti účtu. Dále bude mít k dispozici přehled aktivních účtů na všech databázových serverech a možnost si heslo na vybraném serveru resetovat, kdy mu bude obratem posláno emailem heslo nové.

DB Admina je možno chápat jako pedagoga. DB Admin bude mít na rozdíl od SYS Admina přidělené pouze ty databáze, které může obsluhovat. Bude mít práva pro vytvoření libovolného uživatele a v případě hromadného importu u obou těchto akcí bude muset DB Admin zadat důvod a datum konce platnosti vytvářeného účtu. Důležitým právem bude také řešení žádostí, kde bude moci DB Admin přijaté žádosti povolovat nebo zamítat. Dále bude mít právo nahlédnout do seznamu studentů na vybraném serveru, s tímto souvisí i další seznam účtů, který bude DB Admina informovat o délkách platností jednotlivých účtů.

SYS Admin bude mít stejná práva jako DB Admin s tím rozdílem, že může obsluhovat všechny databáze. Navíc bude mít práva přidávat případně odebírat role v systému a jako hlavní bude obsluhovat správu serverů.

4.1.2 Služby systému

Žádosti

Tato stránka bude mít 4 stavy:

- Účet nebyl doposud vytvořen – v tomto případě bude mít uživatel možnost zažádat o vytvoření účtu společně s vyplněním důvodu žádosti a požadované délky platnosti účtu.
- Účet je aktivní – uživatel má učet vytvořený a přidělená práva k přihlášení
- Účet je zamčen – uživatelův účet je neaktivní, protože mu byly odebrány práva pro přihlášení
- Účet čeká na vyřízení žádosti – uživatelova žádost je evidována a čeká se na její vyřízení

Uživateli budou nabídnuty pouze ty servery, které SYS Admin povolil ve Správě serveru.

Přehled účtů

V této sekci uvidí uživatel všechny servery včetně jejich adres. Dále u nich pozná, zdali má na daném serveru vytvořený uživatelský účet anebo ne. V případě, že zapomene své heslo, bude si zde moci heslo resetovat a následně mu bude e-mailem doručeno heslo nové.

Evidence žádosti

Zde bude Admin moci sledovat jednotlivé žádosti studentů o vytvoření účtů. Společně s loginem žadatele zde uvidí i důvod žádosti a požadovanou délku platnosti. Admin bude mít možnost jednotlivé žádosti buď povolit, nebo zamítnout.

Vytvoření uživatele

Zde bude možné vytvoření jednotlivého uživatele spolu s vyplněním důvodu k vytvoření účtu a také délky platnosti účtu.

Import uživatelů

Systém bude také připraven přijímat soubory vyexportované ze systému Edison do formátu Excel. I zde bude muset uživatel vyplnit důvod k vytváření účtů. Jako první bude muset uživatel požadovaný soubor na server nahrát a systému mu ukáže, kteří uživatelé byli v souboru nalezeni. Vypíše jejich login, jméno a kód oboru, fakulty a ročníku, ve kterém se student nachází. Až po prohlédnutí seznamu uživatel potvrdí import. Systém vytvoří uživatele a informuje o výsledku importu, konkrétně mu vypíše, které účty se podařilo, případně nepodařilo vytvořit.

Seznam uživatelů

Zde bude k dispozici přehled uživatelů na serverech spolu s informací, zdali má uživatel právo pro připojení či nikoliv. Admin bude mít možnost účty mazat jednotlivě nebo hromadně pomocí checkboxu. Dále je zde možnost uživateli odebrat případně přidělit práva pro připojení. V případě uzamčení uživatelova účtu (několikrát špatně zadané heslo) je zde možnost daný účet odblokovat, následně odemknout a poslat uživateli e-mailem nové heslo.

Seznam expirací účtů

Podobný seznam jako seznam účtů, avšak zde bude Admin sledovat platnost jednotlivých účtů a buďto propadlé účty mazat, anebo jim prodlužit platnost.

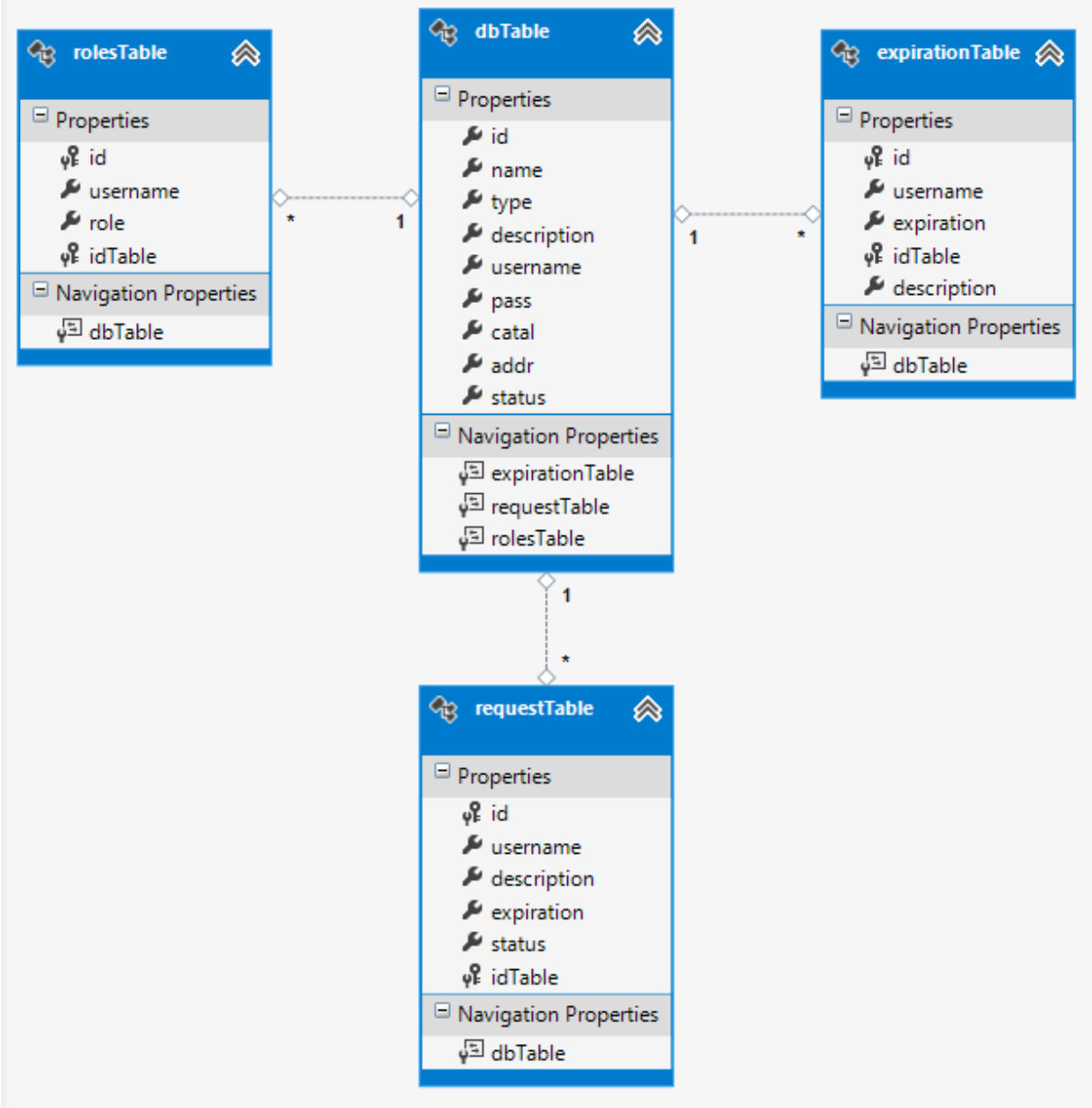
Správa DB Adminů

Správa DB Adminů bude sloužit pro tvorbu rolí v systému, konkrétně zde bude možnost uživateli přidělit buď právo SYS Admin a nebo DB Admin, kdy je potřeba ještě vybrat server, který má budoucí DB Admin obsluhovat.

Správa serverů

Správa serverů je nejdůležitější částí systému. SYS Admin zde bude přidávat nové servery a mazat staré. Dále zde bude možnost zamknout server, což způsobí, že daný server nebude studentovi nabídnut při vytváření žádosti. Důležitou vlastností této stránky bude také informace o stavu serveru, kdy podle příslušné ikony SYS Admin zjistí, zdali je server dostupný nebo ne. Jako poslední zde bude možnost měnit parametry pro připojení k serveru.

4.2 Datová analýza



Obrázek 4.1: Datový model

Popis tabulek:

1. **dbTable** – evidence databází
2. **expirationTable** – evidence platností jednotlivých účtů
3. **requestTable** – evidence žádostí o vytvoření účtů
4. **rolesTable** – evidence rolí SYS Admin a DB Admin

Tabulka 4.1: dbTable – datová analýza

dbTable					
Název	**Typ**	**Velikost**	**Klíč**	**Null**	**Popis**
Id	Integer		PK	Not null	Identifikace záznamu
Name	Varchar	50		Not null	Jméno databáze
Type	Varchar	100		Not null	Typ databáze
Description	Varchar	500		Not null	Popis
Username	Varchar	100		Not null	Přihlašovací jméno
Pass	Varchar	100		Not null	Heslo
Catal	Varchar	150		Not null	Primární katalog
Addr	Varchar	150		Not null	Adresa databáze
Status	Varchar	50		Not null	Stav databáze

Tabulka 4.2: expirationTable – datová analýza

expirationTable					
Název	**Typ**	**Velikost**	**Klíč**	**Null**	**Popis**
Id	Integer		PK	Not null	Identifikace záznamu
Username	Varchar	50		Not null	Uživ. jméno
Expiration	Date			Not null	Datum konce platnosti
idTable	Integer		FK	Not null	ID databáze
Description	Varchar	1000		Not null	Popis

Tabulka 4.3: requestTable – datová analýza

expirationTable					
Název	**Typ**	**Velikost**	**Klíč**	**Null**	**Popis**
Id	Integer		PK	Not null	Identifikace záznamu
Username	Varchar	50		Not null	Uživ. jméno žadatele
Description	Varchar	1000		Not null	Popis žádanky
Expiration	Date			Not null	Datum konce platnosti
Status	Varchar	50		Not null	Stav žádanky
idTable	Integer		FK	Not null	ID žádané databáze

Tabulka 4.4: rolesTable – datová analýza

rolestable					
Název	**Typ**	**Velikost**	**Klíč**	**Null**	**Popis**
Id	Integer		PK	Not null	Identifikace záznamu
Username	Varchar	50		Not null	Uživ. jméno admina
Role	Varchar	50		Not null	Přidělená role
idTable	Integer			Nullable	Přidělená databáze

4.3 Návrh uživatelského rozhraní

Uživatelské rozhraní je přesnou kopií designu stránek dbedu.cs.vsb.cz. Důvodem k tomuto výběru byl již zaběhnutý design, který je studentům známý a spojují si ho s databázovými předměty. V pravé části stránek nalezne uživatel sloupec sloužící pro procházení stránek a také přihlašovací box. Vrchní část stránky je statická, zobrazuje pouze údaje a odkazy na jednotlivé stránky školy či katedry. Levá část stránky je nejdůležitější, zobrazují se zde obsahy stránek. Zde uživatel vkládá informace pro zpracování a obsluhuje celou funkčnost systému. Celá stránka pracuje ve dvou jazycích – Čeština a Angličtina. Změna jazyku se týká jak popisků stránek, tak i vyskakovacích oken.

Ukázku designu lze nalézt v přílohách A, B, C a D.

5 Implementace

5.1 Získávání pohledů v jednotlivých tabulkách

Dále bylo potřeba připravit ještě jednotlivé metody pro získávání pohledů z tabulek a také entitní objekty, které představují jednotlivé řádky vybrané tabulky.

5.1.1 Připojení k databázi

Z ukázky lze vidět, že pro zjištění connection stringu je potřeba sáhnout do konfiguračního souboru, kde je tato informace uložena.

```
SQLiteConnection conn = new SQLiteConnection();
conn.ConnectionString =
ConfigurationManager.AppSettings["ConnectionStringSQLite"].ToString();
conn.Open();
```

5.1.2 Získání pohledů

```
public List<DBRoles> GetRoles()
        {
            SqlCommand selectLoginsCommand = new SqlCommand("SELECT
rolesTable.id, rolesTable.username, rolesTable.role, dbTable.name FROM
rolesTable LEFT JOIN dbTable ON rolesTable.idTable = dbTable.id", conn);
;
            SqlDataAdapter dataAdapter = new SqlDataAdapter();
            DataTable dt = new DataTable();
            List<DBRoles> list = new List<DBRoles>();

            dataAdapter.SelectCommand = selectLoginsCommand;
            dataAdapter.Fill(dt);

            if (dt != null && dt.Rows.Count > 0)
            {
                for (int i = 0; i < dt.Rows.Count; i++)
                {
                    list.Add(new DBRoles(
                        Convert.ToInt32(dt.Rows[i].ItemArray[0]),
                        dt.Rows[i].ItemArray[1].ToString(),
                        dt.Rows[i].ItemArray[2].ToString(),
                        dt.Rows[i].ItemArray[3].ToString()
                        ));
                }
            }
            return list;
        }
```

5.1.3 Entitní objekt

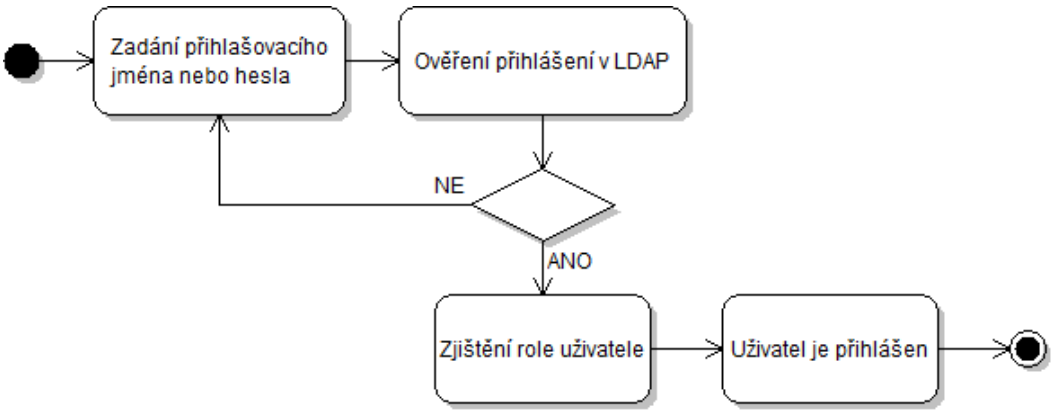
```
public class DBRoles
    {
        public int ID { get; set; }
        public String name { get; set; }
        public String role { get; set; }
        public String db { get; set; }

        public DBRoles(int ID, String name, String role, String db)
```

```
        {
            this.ID = ID;
            this.name = name;
            this.role = role;
            this.db = db;
        }
    }
```

5.2 Přihlašovací systém IS

Jako každý jiný informační systém, musí i tento systém mít funkční přihlášení. Jak již bylo zmiňováno v kapitole návrh řešení, pro správný chod systému je potřeba rozlišovat 3 typy uživatelů (Student, DB Admin a SYS Admin). Pro správu těchto práv slouží sekce Správa DB Adminů, kde může SYS Admin přidělovat práva jiným uživatelům. Pro uložení těchto informací jsem použil tabulku, která ukládá záznam ve tvaru – uživatelské jméno, role, databáze (pouze u DB Admina). Přihlašování funguje na základě ověřování uživatelem zadaných informací na LDAP serveru.



Obrázek 5.1: Activity diagram – průběh přihlášení

5.2.1 LDAP

LDAP (Lightweight Directory Access Protocol) je protokol sloužící pro přístup a ukládání dat na adresářovém serveru. Jak již z názvu vyplývá, LDAP je jakousi odlehčenou verzí, odvozenou od X.500, což je Mezinárodní standard, vyvinutý spolkem International Consultative Commitee of Telephony and Telegraphy, pro formátování elektronických zpráv přenášených prostřednictvím sítě nebo mezi počítačovými sítěmi. Jednotlivé položky jsou ukládány formou záznamu a uspořádány do stromové struktury podle tohoto protokolu. Jedná se většinou o informace o osobách, zaměstnancích, uživatelích (jméno, heslo, e-mail, domovský adresář, telefonní čísla apod). Aplikace funguje na systému klient-server a pro komunikaci se využívá synchronního i asynchronní mód. Součástí LDAP je také autentizace klienta.

Záznam LDAP se skládá z těchto tří prvků:

1. **Distinguished name (DN)** – unikátní název záznamu
2. **Atributy** – popis záznamu
3. **Object classes** – typ záznamu

5.2.2 Implementace přihlášení

Ověření přihlášení v LDAP

Ověření v LDAP [6] má dvě fáze, v první fázi je hledán uživatel v adresáři LDAP. Tento proces ještě nevyužívá uživatelova hesla, tudíž se používá pro získání informací anonymní přístup. Pokud je uživatel nalezen, uloží se DN záznam daného uživatele. Následně je tento záznam použit pro autentifikaci uživatele. Pokud se přihlášení provede úspěšně, vrací funkce hodnotu true, pokud je přihlašovací heslo nesprávné nebo uživatel nebyl nalezen, vyvolá se výjimka a funkce vrací hodnotu false.

Implementace:

```
public bool Authenticate(string userName, string password)
    {
        LdapConnection connection = new LdapConnection(new
LdapDirectoryIdentifier("ldap.vsb.cz", 0x27c))
        {
            SessionOptions = { SecureSocketLayer = true },
            AuthType = AuthType.Anonymous
        };
        using (connection)
        {
            try
            {
                connection.Bind();
            }
            catch
            {
                return false;
            }
            try
            {
                SearchRequest request = new SearchRequest("",
string.Format("(&(objectClass=Person)(uid={0}))", userName), SearchScope.Subtree, new
string[0]);
                SearchResponse response = (SearchResponse)connection.SendRequest(request);
                if (response.Entries.Count == 0)
                {
                    return false;
                }
                SearchResultEntry entry = response.Entries[0];
                string distinguishedName = entry.DistinguishedName;
                connection.Credential = new NetworkCredential(distinguishedName, password);
                connection.AuthType = AuthType.Basic;
                connection.Bind();
                return true;
            }
            catch (Exception)
            {
                return false;
            }
        }
    }
```

Ověření uživatelovy role

Pokud je přihlášení provedeno úspěšně, je potřeba zjistit roli uživatele. Role se zjišťuje pomocí SQL selectu, kdy se vrací první nalezený záznam pro dané uživatelské jméno. Tímto se zajistí, že v případě, kdy by uživatel měl přidělené obě role (DB Admin i SYS Admin), bude uživatelovi vždy nastavena role s větší prioritou, tedy SYS Admin.

SQL Select

```
"SELECT rolesTable.id, rolesTable.username, rolesTable.role, dbTable.name FROM rolesTable
LEFT JOIN dbTable ON rolesTable.idTable = dbTable.id WHERE rolesTable.username=@username
ORDER BY rolesTable.role DESC"
```

Implementace:

```
protected void LoginClick(object sender, EventArgs e)
    {
        if (Authenticate(login, pass))
        {
            mainConn.Connect();
            List<DBRoles> roleDetail = mainConn.GetRoleDetail(login);
            if (roleDetail.Count == 0)
            {
                base.Session["role"] = "student";
            }
            else
            {
                base.Session["role"] = roleDetail[0].role;
            }
            base.Session["logged"] = login;
            mainConn.Destroy();
            LoggedAsLbl.Text = string.Concat(new object[] { "Logged as ",
base.Session["logged"], "(", base.Session["role"], ")" });
            base.Response.Redirect(base.Request.Url.ToString());
        }
        else
        {
            loginFailed.Visible = true;
        }
    }
```

Ochrana proti neoprávněnému přístupu

V případě, kdyby se uživatel s nižší prioritou pokoušel o neautorizovaný přístup ke stránce s větší prioritou, dojde k jeho automatickému přesměrování na úvodní stránku. Jako první se kontroluje, zdali je uživatel vůbec přihlášen. Pokud není, je automaticky přesměrován na úvodní stránku, pokud uživatel přihlášen je, přichází na řadu kontrola jeho role. Pokud uživatelova role nesouhlasí s požadavky dané stránky, je uživatel opět přesměrován na úvodní stránku.

Implementace (příklad pro přístup na Správu serverů)

```
   if (Session["role"] != null)
        {
            if (Session["role"].ToString() != "sysadmin")
            {

base.Response.Redirect(ConfigurationManager.AppSettings["DefaultPage"].ToString());
            }
        }
        else
        {

base.Response.Redirect(ConfigurationManager.AppSettings["DefaultPage"].ToString());
        }
```

Obrázek 5.2: Správa DB adminů

Nabídka jen povolených stránek pro danou roli

Při vytváření menu je potřeba, aby byly uživateli nabídnuty jen ty stránky, ke kterým má mít přístup. Toho je docíleno přímo při načítání podstránky s menu. Jako první se zjišťuje, jakou má uživatel roli a podle této role se následně zobrazují jednotlivé položky menu.

Implementace (částečná)

```
if (base.Session["logged"] != null)
        {
            LoginPhl.Visible = false;
            LogoutPhl.Visible = true;
            LoggedAsLbl.Text = string.Concat(new object[] { "Logged as: </br></br> ",
base.Session["logged"], "(", base.Session["role"], ")" });
            if (base.Session["role"].ToString() == "sysadmin")
            {
                SendRequestMenu.Visible = false;
                AccountsInfoMenu.Visible = true;
                ...
            }
            if (base.Session["role"].ToString() == "dbadmin")
            {
                SendRequestMenu.Visible = false;
                AccountsInfoMenu.Visible = true;
                ExpiredAccountsMenu.Visible = true;
                ...
            }
            if (base.Session["role"].ToString() == "student")
            {
                SendRequestMenu.Visible = true;
                AccountsInfoMenu.Visible = true;
                ...
            }
        }
        else
        {
            LoginPhl.Visible = true;
            LogoutPhl.Visible = false;
            SendRequestMenu.Visible = false;
            ...
        }
```

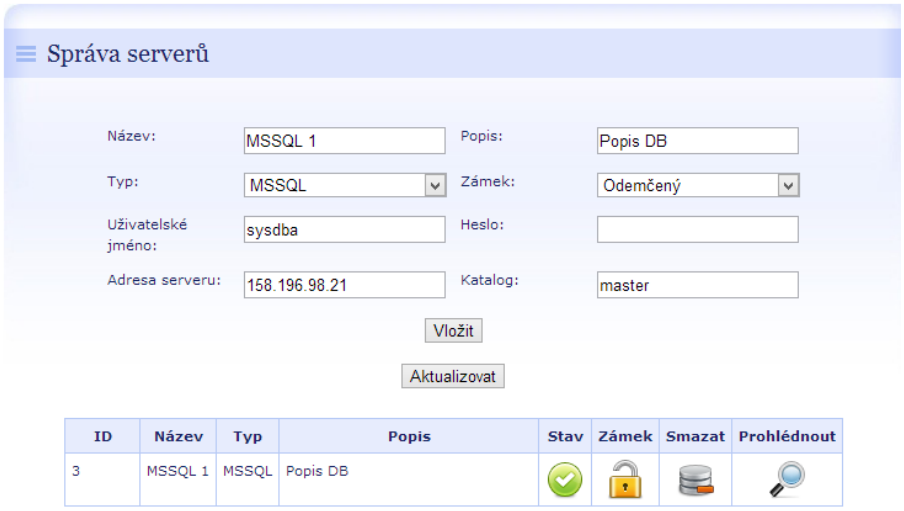
5.3 Administrace databázových systémů

Aby bylo možno spravovat více databázových systémů, je potřeba je nějakým způsobem evidovat. Pro řešení tohoto problému je vytvořena tabulka, kde jsou tyto údaje ukládány.

Evidované údaje databázového serveru:

1. Název
2. Popis
3. Typ
4. Uživatelské jméno
5. Heslo
6. Adresa serveru
7. Katalog

Některé údaje jsou pouze informační (jméno a popis), avšak ostatní údaje jsou potřebné pro úspěšné připojení k databázi. Pro zachování bezpečnosti, je pro ukládání hesla pro přístup k databázi využito 3DES šifrování na úrovni aplikace. Při vkládání záznamu databáze je položka s heslem zašifrována, při čtení je zpětně dešifrována.



Obrázek 5.3: Administrace databázových serverů

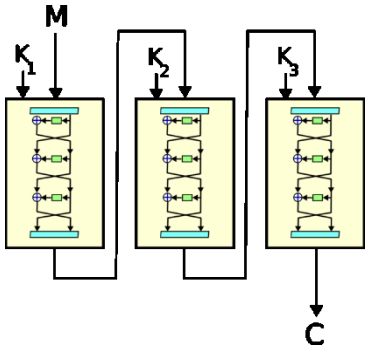
5.3.1 3DES šifrování

Triple DES [4] (také označován jako TDES nebo 3DES) je šifra blokového typu, která je založena na šifrování původního DES (Data Encryption Standard). Rozdíl mezi těmito způsoby šifrování lze odvodit z nového značení 3DES – šifrování se aplikuje pomocí DES 3 krát, tím se značně zvyšuje odolnost vůči útokům a následnému prolomení. Původní DES má délku klíče 53 bitů, což bylo z hlediska bezpečnosti nedostatečné a hrozily útoky hrubou silou tzv. Brute force attack. K řešení tohoto problému se nabízelo jednoduché řešení a to Triple DES díky většímu klíči bez jakékoliv nutnosti řešit nový algoritmus šifrování. Nevýhodou tohoto šifrovacího algoritmu je jeho pomalost.

Jak již bylo řečeno, Triple DES je trojnásobnou aplikací šifry DES. I když by bylo jednodušší pro šifrování opakovat tři šifrovací aplikace pouze po sobě (tzv. varianta EEE), v praxi se používá varianta označována EDE, která na rozdíl od EEE nejprve zašifruje vkládaný text prvním klíčem, poté výsledek dešifruje druhým klíčem a nakonec tento výsledek ještě zašifruje třetím klíčem.

$$C = E\left(K_3, D\left(K_2, E(K_1, M)\right)\right)$$

Ve vzorci je E šifrovací operace DES, D je dešifrovací operace K1, K2 a K3 jsou jednotlivé podklíče (dohromady tvoří TDES klíč) M je vložený text a C je šifrovaný text.



Obrázek 5.4: Trojnásobná aplikace šifry DES – 3DES

Implementace (šifrování):

```
TripleDESCryptoServiceProvider des = new TripleDESCryptoServiceProvider();
MD5CryptoServiceProvider md5 = new MD5CryptoServiceProvider();
byte[] byteHash, byteBuff;
byteHash = md5.ComputeHash(ASCIIEncoding.ASCII.GetBytes("dbedu"));
md5 = null;
des.Key = byteHash;
des.Mode = CipherMode.ECB;
byteBuff = Convert.FromBase64String(dataToCrypt);
String decryptedData =
ASCIIEncoding.ASCII.GetString(des.CreateDecryptor().TransformFinalBlock(byteBuff
, 0, byteBuff.Length));
```

Implementace (dešifrování):

```
TripleDESCryptoServiceProvider des = new TripleDESCryptoServiceProvider();
MD5CryptoServiceProvider md5 = new MD5CryptoServiceProvider();
byte[] byteHash, byteBuff;
byteHash = md5.ComputeHash(ASCIIEncoding.ASCII.GetBytes("dbedu"));
md5 = null;
des.Key = byteHash;
des.Mode = CipherMode.ECB;
byteBuff = ASCIIEncoding.ASCII.GetBytes(pass);
String encryptedData =
Convert.ToBase64String(des.CreateEncryptor().TransformFinalBlock(byteBuff, 0,
byteBuff.Length));
```

5.3.2 Generika - třída a použití rozhraní

První implementace systému počítala pouze s nasazením na Microsoft SQL Server a Oracle Database 11g. Pro vytvoření instance tedy stačilo rozlišit pouze dva typy připojení. V průběhu implementace jsem se rozhodl dosavadní řešení vylepšit a systém připravit pro dodatečnou implementaci jiných databázových systémů (MySQL, DB2, Postgres apod.). S dosavadním řešením, by tato implementace znamenala značný problém pro zaručení funkčnosti. Pokud by vývojář chtěl například přidat 5 nových systémů, musel by na každé stránce rozlišovat dalších 5 nových systémů. Tento problém však vyřeší použití generiky. Vytvořil jsem tedy rozhraní, které mi definuje potřebné funkce a metody pro správnou funkčnost systému. V tomto novém řešení je potřeba pouze zjistit typ databáze ze záznamu databáze a následně ho vložit do funkce Type.GetType(), která mi vytvoří instanci objektu podle zadaného typu.

Implementace rozhraní:

```
public interface IConnections
    {
        bool Connect();
        bool CreateLogin(String login, String pass);
        bool DeleteLogin(String login);
        bool GrantConnect(String login);
        bool DenyConnect(String login);
        List<String> GetLogins();
        List<Logins> GetLoginsPermissions(String dbname);
        bool CheckLogin(String login);
        bool CheckPermission(String login);
        bool CheckLock(String login);
        bool UnlockAccount(String login);
        bool PasswordReset(String login, String pass);
        bool Destroy();
    }
```

Vytvoření objektu:

```
IConnections conn = (IConnections)Activator.CreateInstance(Type.GetType("Connections." + db.type + ",Connections"), db.connection);
```

5.3.3 Výběr a tvorba adekvátních connection stringů

Jak již bylo psáno výše, pro vytvoření připojení na databázi, je potřeba 4 údajů (uživatelské jméno, heslo, adresa a katalog). Ovšem tyto údaje samotné připojení na databází nevytvoří. Jako připojovací údaj slouží tzv. Connection String, neboli připojovací řetězec obsahující údaje potřebné k připojení. Avšak každý systém má jiný formát svého Connection Stringu. Proto je zde opět potřeba rozlišit typ databáze a následně vybrat správnou šablonu Connection Stringu. Pro vytváření finálního Connection Stringu byla použita funkce String.Format(). Za hodnoty {číslo} vkládám pomocí této funkce připojovací údaje. Connection String se vytváří automaticky při volání konstruktoru pro entitu databáze.

Příklad šablony Connection Stringu

```
<add key="MssqlString" value="Data Source={3};Initial Catalog={2};Persist Security Info=True;User ID={0};Password={1};Connection Timeout=5;"/>
```

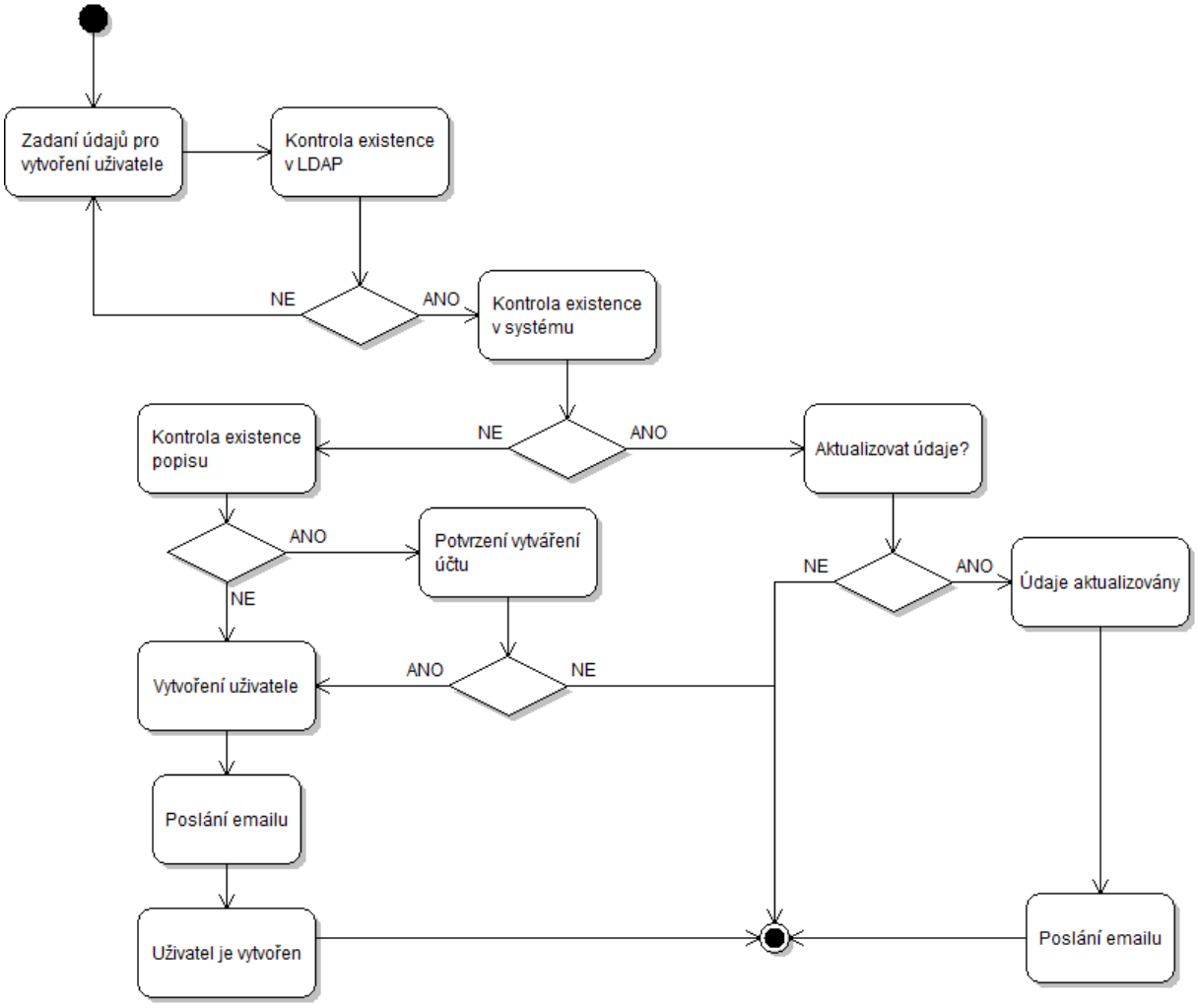
Generování Connection Stringu

```
this.connection = String.Format(ConfigurationManager.AppSettings[type + "String"],username, pass, catalog, address);
```

5.4 Administrace databázových uživatelů

Nejdůležitějším prvkem tohoto systému je administrace databázových uživatelů. Veškeré SQL příkazy pro manipulaci s účty na serverech Microsoft SQL a Oracle jsou obsaženy v přílohách E, F, G a H. Pro mazací příkazy je nutno připomenout, že před samotným vymazání uživatele je potřeba natvrdo ukončit všechny jeho činnosti vykonávané na serveru. Systém nabízí tři možnosti vytvoření účtu:

1. Na základě žádosti uživatele
2. DB nebo SYS Admin jej vytvoří samostatně
3. DB nebo SYS Admin jej importuje z Excel souboru, vygenerovaným ze systém Edison



Obrázek 5.5: Activity diagram – průběh vytvoření uživatele

5.4.1 Kontrola existence v LDAP

Díky této funkci se zamezí, aby byli na serverech vytvářeni uživatelé, kteří nejsou součásti LDAP. V případě, že uživatel není nalezen v LDAP, systém na to upozorní a průběh vytváření účtu zastaví. Implementace je jednoduchá, pro tento účel je vytvořena třída LDAP_Person, která přijímá v konstruktoru pouze argument login typu String. Následně zjišťuje pomocí tohoto loginu zbylé informace o uživateli. Pokud uživatel nebyl nalezen, atributy tohoto objektu budou mít hodnoty null.

Implementace konstruktoru objektu LDAP_Person:

```
public LDAP_Person(String login)
        {
            LdapConnection con = new LdapConnection(new
LdapDirectoryIdentifier("ldap.vsb.cz", 636));
            con.SessionOptions.SecureSocketLayer = true;
            con.AuthType = AuthType.Anonymous;
            using (con)
            {
                try
                {
                    SearchRequest request = new SearchRequest(
                    "",
                    String.Format("(&(objectClass=Person)(uid={0}))", login),
                    System.DirectoryServices.Protocols.SearchScope.Subtree
                    );
                    SearchResponse response =
(SearchResponse)con.SendRequest(request);
                    if (response.Entries.Count == 0)
                    {
                        //NEBYL NALEZEN
                    }
                    else
                    {
                        SearchResultEntry entry = response.Entries[0];
                        string dn = entry.DistinguishedName;
                        string path = "LDAP://ldap.vsb.cz:389/" + dn;
                        DirectoryEntry dentry = new DirectoryEntry(path);
                        dentry.AuthenticationType = AuthenticationTypes.Anonymous;
                        DirectorySearcher search = new DirectorySearcher(dentry);
                        SearchResult result = search.FindOne();

                        this.login = result.Properties["cn"][0].ToString();
                        this.fname = result.Properties["givenname"][0].ToString();
                        this.lname = result.Properties["sn"][0].ToString();
                        this.mail = result.Properties["mail"][0].ToString();
                    }
                }
                catch
                {

                }
            }
        }
```

Implementace:

```
if (new LDAP_Person(login).login == null)
        {
            outputLabel.Text = alert3; //VÝPIS CHYBY
        }
```

5.4.2 Kontrola existence účtu na serveru

V případě, kdy Admin vytváří uživatele samostatně, se dále kontroluje, zdali už tento uživatel nemá vytvořený účet. Pokud ano, dojde k upozornění Admina, ve kterém se vypíše do kdy a za jakým účelem má uživatel vytvořený účet a nabídne mu možnost tyto údaje aktualizovat. V tomto případě se pošle uživateli e-mail s informacemi o aktualizaci údajů o vytvoření účtu.

5.4.3 Kontrola existence popisu

Systém také při vytváření uživatele kontroluje, zdali už v systému neexistuje stejný popis k vytvoření uživatele. Pokud je stejný popis nalezen, systém na to Admina upozorní a požaduje potvrzení pro pokračování ve vytváření účtu.

5.4.4 Samotné vytvoření uživatele

Pokud jsou všechny předešlé podmínky splněny, přichází na řadu vytvoření uživatele. Tento proces ještě doprovází generování hesla na základě předem zadané délky v konfiguračním souboru systému. Opět se zde využívá již zmíněné generiky pro vytvoření objektu připojení k databázi a následné zavolání funkce pro vytvoření uživatele, která zároveň zjišťuje, zdali se uživatele podařilo vytvořit či nikoliv. V případě úspěšného vytvoření uživatele systém ještě vloží záznam o vytvoření uživatele společně s popisem a datem konce platnosti a odešle e-mail se všemi potřebnými údaji k uživateli.

Implementace generátoru hesel:

```
public String GeneratePassword()
    {
        string allowedChars =
"abcdefghijkmnopqrstuvwxyzABCDEFGHJKLMNOPQRSTUVWXYZ0123456789";
        int passwordLength =
System.Convert.ToInt32(ConfigurationManager.AppSettings["PasswordLength"].ToString());
        char[] chars = new char[passwordLength];
        Random rd = new Random();

        for (int i = 0; i < passwordLength; i++)
        {
            chars[i] = allowedChars[rd.Next(0, allowedChars.Length)];
        }

        return new string(chars);
    }
```

Implementace vytvoření uživatele:

```
        if (mainConn.CheckDescription(reason, selectedServer) == false)
           {
                String pass = GeneratePassword();

                conn = (IConnections)Activator.CreateInstance(Type.GetType("Connections." +
selectedServer + ",Connections"), serverID);
                conn.Connect();
                if (conn.CreateLogin(login, pass))
                {
                    mainConn.InsertExpiration(login, endDate, reason);
                    SendMail(new LDAP_Person(Session["logged"].ToString()), new
LDAP_Person(login), mailTextOK, mailSubjectOK, pass, DBAddress, endDate);
                    string[] strArray = message1.Split(new char[] { '#' });
                    outputLabel.Text = strArray[0] + login + strArray[1];
                }
                else
                {
                Expirations exp = mainConn.GetExpirationDescription(selectServer, login);
                if (exp != null)
                    {
                        string[] strArray2 = message2.Split(new char[] { '#' });
                        outputLabel.Text = strArray2[0] + login + strArray2[1] +
exp.description + strArray2[2] + endDate + "!";
                        extendButton.Visible = true;
                    }
                    else
                    {
                        outputLabel.Text = alert5;
                    }
                }
                conn.Destroy();
                confirmButton.Visible = false;
            }
            else
            {
                outputLabel.Text = alert4;
                confirmButton.Visible = true;
            }
        }
        mainConn.Destroy();
```

5.4.5 Import uživatelů

Na rozdíl od samostatného vytváření uživatelů, hromadné vytváření přijímá jako vstupní data Excel soubor s vygenerovanými údaji pomocí systému Edison. Celý průběh vytváření je stejný jako u samotného vytváření s tím rozdílem, že zde se operace provádějí cyklicky. Admin si musí nejprve pomocí systému Edison nechat vygenerovat svou skupinu studentů do formátu Excel. Poté si tento soubor vybere a provede import dat a v tabulce se mu ukážou údaje o studentech, které chce importovat. Jako poslední musí uživatel tento seznam studentů potvrdit a tím se zahájí vytváření uživatelů. Výsledkem je další tabulka, která Admina informuje o úspěchu případně neúspěchu vytváření jednotlivých účtu. Avšak jediný problém je v tom, že vložený soubor nelze číst přímo od uživatele, soubor se musí prvně nahrát na server, data se přečtou a následně se soubor smaže.

Implementace:

```
public List<StudentsImport> getImportedList(string file)
    {
        List<StudentsImport> list = new List<StudentsImport>();
        OleDbConnection connection = new
OleDbConnection("Provider=Microsoft.Jet.Oledb.4.0;Data Source=" + file + ";Extended
Properties=Excel 8.0");
        string cmdText = "SELECT * FROM [StudentsList$]";
        OleDbCommand selectCommand = new OleDbCommand(cmdText, connection);
        DataTable dataTable = new DataTable();
        new OleDbDataAdapter(selectCommand).Fill(dataTable);
        int num = 0;
        foreach (DataRow row in dataTable.Rows)
        {
            if ((num > 8) && (row[2].ToString().Length > 0))
            {
                list.Add(new StudentsImport(row[2].ToString(), row[3].ToString(),
row[4].ToString()));
            }
            num++;
        }
        return list;
    }
```

5.4.6 Posílání emailu

Každá stránka má jinak definovanou metodu pro posílání e-mailu. Konkrétně se jedná o část metody s argumenty a následně o tělo zprávy. Opět je zde využito již zmíněného objektu LDAP_Person, který v tomto případě slouží pro zjištění údajů odesilatele a příjemce.

Implementace

```
void SendMail(LDAP_Person from, LDAP_Person to, String msg, String subject, String
pass, String address, DateTime date)
    {
        try
        {
            MailMessage message = new MailMessage();
            SmtpClient smtp = new SmtpClient("smtp.vsb.cz");

            message.To.Add(to.mail);
            message.From = new
MailAddress(ConfigurationManager.AppSettings["Mail"].ToString());
            message.Subject = subject;
            message.IsBodyHtml = true;
            message.Body = msg ;
            smtp.Port = 25;
            smtp.Send(message);

        }
        catch
        {
        }
    }
```

6 Závěr

Cílem této práce bylo vytvořit informační systém, který se bude používat pro administraci účtů na katedrových databázových systémech. Řešení tohoto systému bylo rozloženo do několika stěžejních bodů, které bylo potřeba postupně splnit.

Úvaha nad problematikou byla zcela jasná, dosavadní řešení představuje pouze jednoduché řešení administrace, bez možnosti sledování expirací účtů, bez možnosti importu z vygenerovaných *.xls souborů importovaných systémem Edison.

Dále bylo potřeba si stanovit, které systémy se pro vývoj informačního systému použijí. Jelikož byla vybrána technologie ASP.Net, tak jasnou volbou byl také Windows Server, na kterém tento systém poběží. Hlavní výhodou použití těchto technologií je, že oproti dosavadnímu řešení, bude možno obsluhovat databáze i z prostředí mimo školu. To zajistí právě vytvořené webové rozhraní informačního systému.

Jako další přišly na řadu Funkční a Datová analýza, kde bylo potřeba se ujistit, jaké bude mít systém možnosti, k čemu bude sloužit a jaké role uživatelů ho budou obsluhovat. Vše bylo detailně připraveno pro bezproblémové zavedení implementace systému. Datová analýza zahrnovala pouze 4 tabulky, potřebné pro správný chod systému a evidenci dat.

Jako poslední zůstalo řešení implementace, které zahrnovalo spoustu problematiky. Jako první bylo potřeba zajistit, aby systém používal ověření přihlášení pomocí školního LDAP. Dále bylo potřeba připravit generické rozhraní a k němu příslušné třídy reprezentující daný typ databázového serveru (Microsoft SQL, Oracle). To umožnilo dynamické vytváření objektů na základě typu právě obsluhované databáze. Poté přišla na řadu implementace samotných webových stránek a zajištění jejich funkcionality, což zahrnuje práci s databázovými účty, zobrazovaní pohledů na základě získaných dat z databáze, posílání e-mailu či ověřovací funkce jednotlivých textboxů.

Všechny tyto body byly splněny a systém je plně funkční. Systém v období vývoje běžel pouze na virtuálním stroji, kde probíhalo testování více osobami. Toto řešení umožňovalo detekci chyb a jejich následné vyladění ke správné funkčnosti. Jsem přesvědčeno o tom, že systém bude katedře velmi nápomocen a značně jim ulehčí práci spojenou s vytvářením účtu pro databázové předměty, apod.

Použitá literatura

[1] BRYLA, Bob a Kevin LONEY. *Mistrovství v Oracle Database 11g*. Vyd. 1. Brno: Computer Press, 2009, 700 s. ISBN 978-80-251-2189-4.

[2] BELLINASO, Marco a Kevin LONEY. *Webové programování v ASP.NET 2.0: problém, návrh, řešení*. Vyd. 1. Překlad Lukáš Krejčí. Brno: Computer Press, 2007, 648 s. ISBN 978-80-251-1893-1.

[3] STANEK, William R a Kevin LONEY. *Microsoft SQL Server 2012: kapesní rádce administrátora*. 1. vyd. Překlad Lukáš Krejčí. Brno: Computer Press, 2013, 544 s. Microsoft (Computer Press). ISBN 978-80-251-3797-0.

Internetové zdroje

[4] Triple DES - Wikipedia. *Wikipedie, otevřená encyklopedie* [online]. [cit. 2013-04-24]. Dostupné z: http://cs.wikipedia.org/wiki/3DES

[5] IBM DB2 - Wikipedia. *Wikipedie, otevřená encyklopedie* [online]. [cit. 2013-04-24]. Dostupné z: http://cs.wikipedia.org/wiki/IBM_DB2

[6] LDAP - Wikipedia. *Wikipedie, otevřená encyklopedie* [online]. [cit. 2013-04-24]. Dostupné z: http://cs.wikipedia.org/wiki/LDAP

[7] Servery Sun - Integrované systémy - Oracle. *Oracle Česká republika | Hardware and Software, Engineered to Work Together* [online]. [cit. 2013-04-24]. Dostupné z: http://www.oracle.com/cz/products/servers/overview/index.html

[8] Databázový server | Software pro správu dat | Microsoft SQL Server 2008. *Microsoft Home Page | Devices and Services* [online]. [cit. 2013-04-24]. Dostupné z: http://www.microsoft.com/sqlserver/cs/cz/product-info.aspx

[9] Technologie ASP.NET - přehled. *Microsoft Home Page | Devices and Services* [online]. [cit. 2013-04-24]. Dostupné z: http://technet.microsoft.com/cs-cz/library/cc728044(v=ws.10).aspx

[10] Oracle Documentation. *Oracle | Hardware and Software, Engineered to Work Together*[online]. [cit. 2013-04-24]. Dostupné z: http://docs.oracle.com/

[11] PostgreSQL - Wikipedie. *Wikipedie, otevřená encyklopedie* [online]. [cit. 2013-04-26]. Dostupné z: http://cs.wikipedia.org/wiki/PostgreSQL

[12] MySQL - Wikipedie. *Wikipedie, otevřená encyklopedie* [online]. [cit. 2013-04-26]. Dostupné z: http://cs.wikipedia.org/wiki/Mysql

Seznam příloh

Součástí BP/DP je CD/DVD.

Adresářová struktura přiloženého CD/DVD:

/doc – elektronická verze bakalářské práce

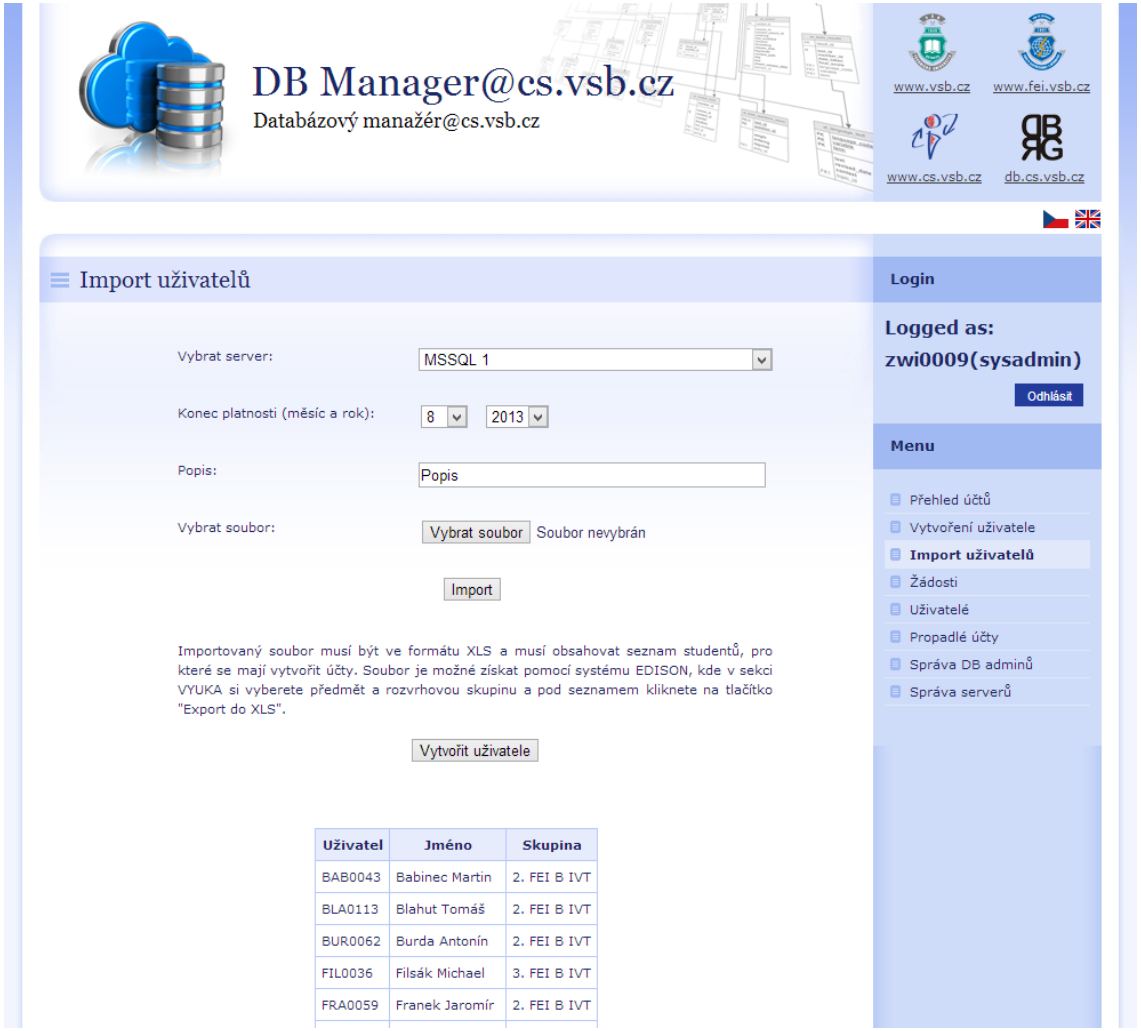
/project – projekt informačního systému

/sql – potřebné SQL příkazy pro nastavení databází

Příloha A: *GUI – Základní okno*

Příloha B: *GUI – Přihlášení a přehled účtů*

Příloha C: *GUI – Import uživatelů*



Příloha D: *GUI – Administrace databázových serverů*

Příloha E: Microsoft SQL – Vytvoření uživatele

```
USE [master]
GO
/****** Object:  StoredProcedure [dbo].[CreateUser]    Script Date: 4/28/2013 1:52:56
AM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

-- use master;

ALTER /*ALTER*/ PROCEDURE [dbo].[CreateUser] @p_UserName VARCHAR(20),@p_passwd
VARCHAR(20)
AS
DECLARE @cmd NVARCHAR(200)

SET @cmd = 'CREATE DATABASE ' + @p_UserName + ';';
EXEC sp_executesql @cmd;

SET @cmd = 'CREATE LOGIN ' + @p_UserName + ' WITH PASSWORD = ''' + @p_passwd + ''',
CHECK_POLICY = OFF;';
EXEC sp_executesql @cmd;

SET @cmd = 'ALTER LOGIN ' + @p_UserName + ' WITH DEFAULT_DATABASE = ' + @p_UserName +
';';
EXEC sp_executesql @cmd;

SET @cmd = 'USE ' + @p_UserName + ';' + 'CREATE USER ' + @p_UserName + ' FOR LOGIN ' +
@p_UserName + ' WITH DEFAULT_SCHEMA = dbo;'; -- + @p_UserName + ';';
EXEC sp_executesql @cmd;

SET @cmd =  'USE ' + @p_UserName + ';' + 'GRANT CREATE TABLE To ' + @p_UserName + ';';
EXEC sp_executesql @cmd
SET @cmd = 'USE ' + @p_UserName + ';' + 'GRANT ALTER To ' + @p_UserName + ';';
EXEC sp_executesql @cmd
SET @cmd = 'USE ' + @p_UserName + ';' + 'GRANT CONTROL To ' + @p_UserName + ';';
EXEC sp_executesql @cmd

-- it is necessary for bulkload
SET @cmd = 'USE ' + @p_UserName;
EXEC sp_executesql @cmd;
EXEC sp_addsrvrolemember @p_UserName, 'bulkadmin';
```

Příloha F: Microsoft SQL – Smazání uživatele

```
USE [master]
GO
/****** Object:  StoredProcedure [dbo].[DropUser]    Script Date: 4/28/2013 1:53:09 AM
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

ALTER  PROCEDURE [dbo].[DropUser] @p_UserName VARCHAR(20)
AS
DECLARE @cmd NVARCHAR(200);
DECLARE dropCurs CURSOR FOR SELECT session_id FROM sys.dm_exec_sessions WHERE
login_name = @p_UserName;
DECLARE @id INT;

OPEN dropCurs;
FETCH NEXT FROM dropCurs INTO @id;
WHILE @@FETCH_STATUS = 0
    BEGIN
       SET @cmd = 'KILL ' + CAST(@id AS NVARCHAR(10)) + ';';
       EXEC sp_executesql @cmd;
    FETCH NEXT FROM dropCurs INTO @id;
END
CLOSE dropCurs;
DEALLOCATE dropCurs;

SET @cmd = 'USE ' + @p_UserName + ';DROP USER ' + @p_UserName + ';';
EXEC sp_executesql @cmd;

SET @cmd = 'DROP LOGIN ' + @p_UserName + ';';
EXEC sp_executesql @cmd;

SET @cmd = 'DROP DATABASE ' + @p_UserName + ';';
EXEC sp_executesql @cmd;
```

Příloha G: Oracle – Vytvoření uživatele

```
create or replace PROCEDURE CreateUser
(
username IN VARCHAR2,
pass IN VARCHAR2
)
AS
BEGIN
DBMS_OUTPUT.PUT_LINE('CREATE USER ' || username || ' IDENTIFIED BY ' || pass || '
DEFAULT TABLESPACE USERS TEMPORARY TABLESPACE TEMP');

EXECUTE IMMEDIATE('CREATE USER ' || username || ' IDENTIFIED BY ' || pass || ' DEFAULT
TABLESPACE USERS TEMPORARY TABLESPACE TEMP');
EXECUTE IMMEDIATE('GRANT CONNECT TO ' || username);
EXECUTE IMMEDIATE('GRANT RESOURCE TO ' || username);
EXECUTE IMMEDIATE('GRANT CREATE VIEW TO ' || username);
EXECUTE IMMEDIATE('GRANT CREATE TABLE TO ' || username);
EXECUTE IMMEDIATE('GRANT SELECT_CATALOG_ROLE TO ' || username);
EXECUTE IMMEDIATE('GRANT SELECT ANY DICTIONARY TO ' || username);
END;
```

Příloha H: Oracle – Smazání uživatele

```
create or replace
PROCEDURE DropUser
(
p_username IN VARCHAR2
)
AS
  CURSOR killCurs IS select sid,serial# from v$session where username =
UPPER(p_username);
  p_sid INT;
  p_serial INT;
BEGIN
  FOR killID IN killCurs LOOP
    DBMS_OUTPUT.PUT_LINE('ALTER SYSTEM KILL SESSION ''' || killID.sid || ',' ||
killID.serial# || ''' IMMEDIATE');
    EXECUTE IMMEDIATE ('ALTER SYSTEM KILL SESSION ''' || killID.sid || ',' ||
killID.serial# || ''' IMMEDIATE');
  END LOOP;
  EXECUTE IMMEDIATE ('DROP USER ' || p_username || ' CASCADE');
END;
```